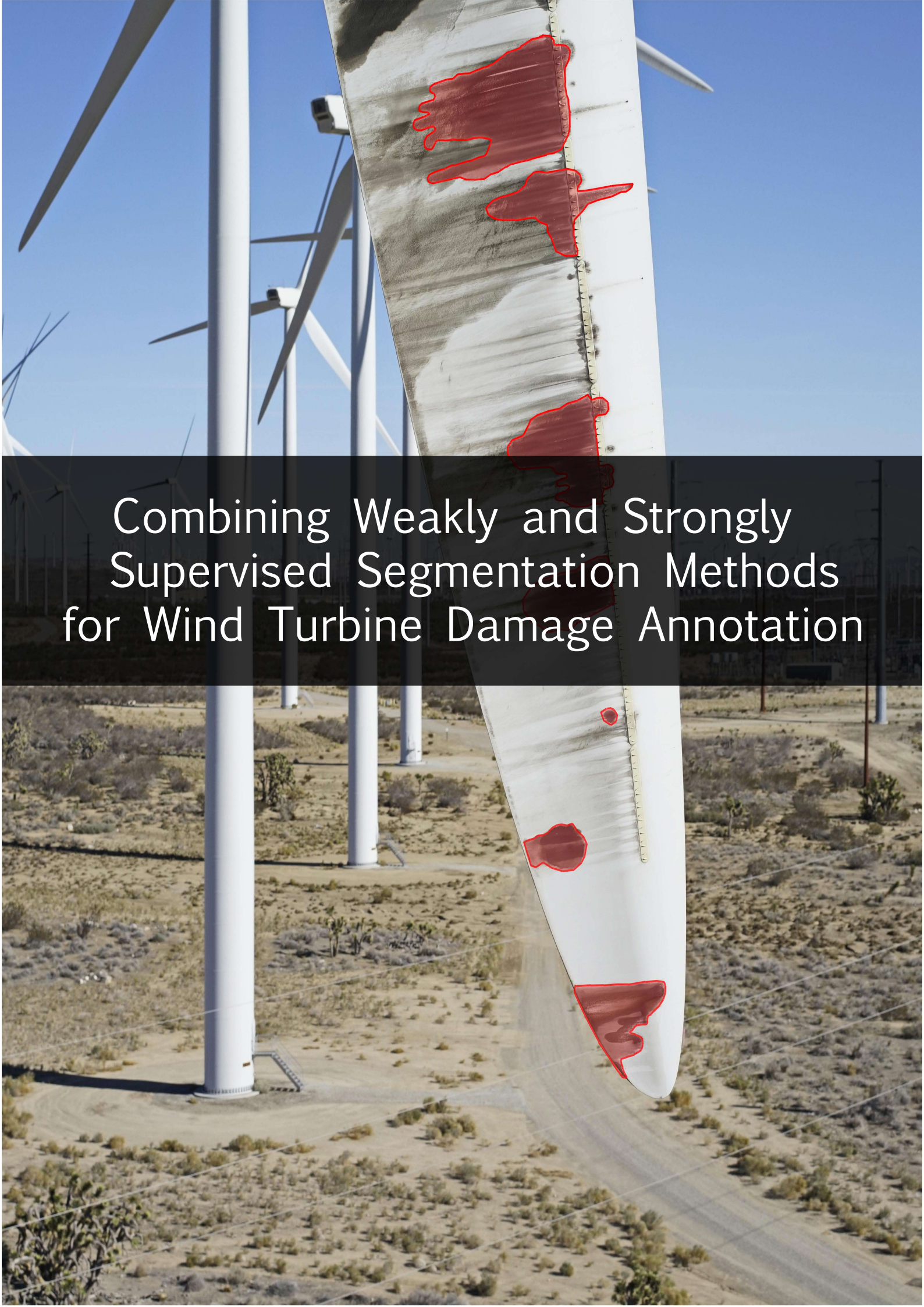


Combining Weakly and Strongly Supervised Segmentation Methods for Wind Turbine Damage Annotation

Combining Weakly and Strongly Supervised Segmentation Methods for Wind Turbine Damage Annotation

Maximilian Crous
10771085

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisors

dhr. dr. Arnoud Visser
Informatics Institute, Faculty of Science
University of Amsterdam
Science Park 904, 1098 XH Amsterdam

Anouk Visser, MSc
Birds.ai
Julianalaan 67A, 2628 BC Delft

June 29th, 2018

Abstract

This thesis investigates the use of image processing methods to assist and accelerate the creation of wind turbine damage annotations. Traditionally, annotations are made manually by drawing polygons, which is a time consuming and labour-intensive process. The general problem of automating such an annotation task is known as instance segmentation. Many algorithms can solve segmentation tasks, yet all require training data annotated with polygons to some extent. To avoid any manual annotations, a combination of Class Activation Mappings (CAMs) and the Mask R-CNN segmentation model is suggested as a viable alternative. General information concerning damage location is captured by CAMs, which is then interpreted as pixel-level data and used to train Mask R-CNN. To the best of our knowledge there have been no works published on this unique combination of weakly and strongly supervised methods. The Mask R-CNN model that was trained for this thesis achieved an average precision of 20% with an IoU threshold of 0.30, doubling the performance of the unrefined CAMs. We hope that this approach will serve as a motivator for future research in combining weakly and strongly supervised methods. The final model enables users to create a polygon that covers wind turbine damage with a certainty of 92% with only two mouse clicks.

Contents

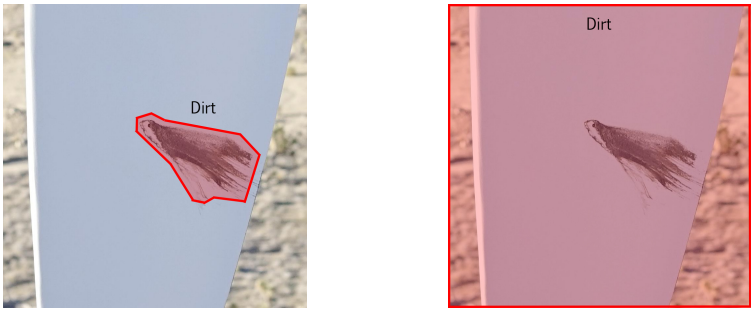
Acknowledgements

I wish to express my sincere thanks to Anouk Visser. Without her guidance, weekly sit-down sessions and willingness to share any required company resources, this paper would have never been accomplished. I would also like to thank Arnoud Visser, who set aside time to provide guidance to both me and other Artificial Intelligence students. I am grateful for the past three years that I have been able to spend at the University of Amsterdam, where I have learned many valuable things and met amazing people. Finally, I would like to thank my family, Lucia, Freek and Mark.

CHAPTER 1

Introduction

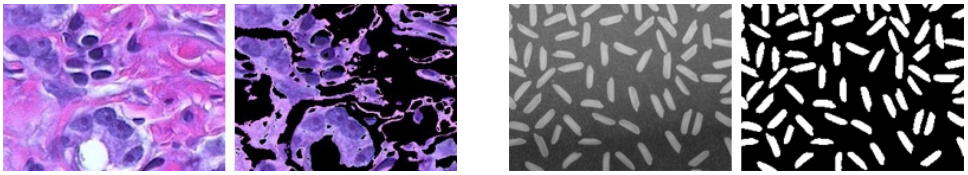
Birds.ai provides companies with insight into the current and future state of company assets. This is achieved by analysing aerial photographs with artificial intelligence to detect asset damage. Birds.ai's recent ventures include the inspection of solar panels, power lines and wind turbines. To provide clients with an accurate description of the discovered damages, a class label and a polygon that encompasses the damage is assigned to each damaged area, as shown in Fig 1.1a. As with many systems that utilise artificial intelligence, a substantial amount of training data is required in order to provide reliable predictions of these labels and polygons. Traditionally, Birds.ai would produce training data by manually creating polygon annotations, which is a time consuming and labour-intensive process. This thesis is an account of an attempt to accelerate and automate this polygon creation process.



(a) Pixel-level label (b) Image-level label

Figure 1.1: Annotation styles

In this context, polygons are used to demarcate a region. In image processing, the general challenge of demarcating a region that pertains to an object or class is called segmentation. Segmentation is a well studied problem that has a large range of practical applications, such as vision systems in autonomous vehicles, medical imaging analysis and evidently asset inspection. Algorithms used in the earliest automated segmentation attempts relied on chosen features and low-level cues such as color and texture [1]. Take, for example, color-based segmentation with K-means clustering [2] and thresholding on grayscale images with Otsu's method [3] in Fig 1.2. Most state of the art approaches have abandoned chosen features in favour of learned features, often by employing some form of convolutional neural networks. SegNet and Mask R-CNN are two of such recently published segmentation networks that have achieved exceptionally high accuracy scores on large-scale data sets COCO and Pascal VOC [4][5]. These models are explicitly mentioned because they demonstrate the difference between the two main fields of recent segmentation research: semantic segmentation and instance segmentation. A semantic segmentation algorithm like SegNet will associate each image pixel with a class label, whereas an instance segmentation algorithm like Mask R-CNN will segment certain individual objects

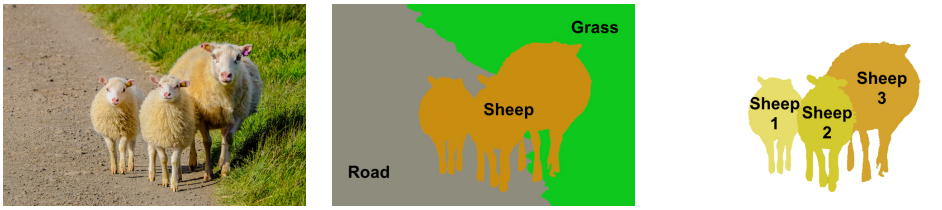


(a) K-means color clustering [2] (b) Otsu's thresholding [3]

Figure 1.2: Dated segmentation methods, images from mathworks[1]

within a scene (see Fig 1.3).
The objects of interest in this thesis are damaged surfaces on wind turbine blades. In the photographs provided by Birds.ai most of the image pixels do not cover damaged blade surfaces and are therefore not of interest. With this in mind, it was decided to only investigate instance segmentation with a Mask R-CNN network. To simulate the constraint of not having expensive manually created polygon annotations, only image-level labels are used to train the instance segmentation network. Thus, the technical goal of this thesis became *to train an instance segmentation network to create polygons around wind turbine damage without using any polygon annotations as training data.*

Because Mask R-CNN and the vast majority of other instance segmentation algorithms require polygons or pixel-level labels as training data, this constraint poses a fundamental problem. Overcoming this challenge demanded a weakly supervised segmentation method, i.e., an algorithm that could derive polygons from image-level labels. To this end, Class Activation Mappings (CAMs) were utilised. These mappings indicate the discriminative image regions used by a classifier to identify the image as a class, effectively transforming image-level labels to pixel-level labels and thereby providing training data for the Mask R-CNN network. The final network trained for this project achieved an average precision of 20% with an IoU threshold of 0.30 when compared to ground truth human annotations and doubled the performance of unrefined CAMs. The considerable performance of this segmentation pipeline, combined with the novel absence of any manually created polygons, renders it an easily applicable and valuable tool for Birds.ai.



(a) Original image (b) Semantic segmentation (c) Instance segmentation

Figure 1.3: Contemporary segmentation methods, images from O'Reilly [2]

[1]https://n l.mathworks.com/discovery/image-segmentation.html
[2]https://www.oreilly.com/ideas/introducing-capsule-networks

CHAPTER 2

Theoretical background

This chapter provides an overview and theoretical investigation of the image processing techniques that were used in this thesis. Each method will be introduced by presenting the problem that it solves.

2.1 Class Activation Mappings

The constraint of having no polygons or pixel-level labels in the training data introduces the demand for a method that can either generate such data or mitigate the reliance on such data. Mitigating the reliance on polygons would mean that the architecture of the instance segmentation network, in this case Mask R-CNN, would have to be modified. Altering such an intricate network seemed like an unrealistic undertaking for the purpose of this work. Instead, a method was chosen that could generate the required data. Naturally, to generate data one must rely on information that specifies, at least to some extent, what exactly is to be generated. Thus, in this context, generating data is taken to mean *converting the image-level labels of damage that are already at our disposal to the required pixel-level labels.*

Weakly supervised segmentation is an emerging technique that is capable of performing such a conversion. The technique has hitherto remained largely unresearched in the computer vision community, with Zhou et al. calling the method "unexplored" in [6] and very few works being dedicated to the subject. Despite its obscurity, there are readily available implementations of weakly supervised segmentation algorithms. One of these is the Class Activation Mappings generator [7]. The code used by the authors was published on Github[1].

Class Activation Mappings or CAMs indicate the discriminative image regions used by a classifier to identify the image as a certain class. This does not only apply to the class with the highest confidence score, but to every class that a classifier was trained on. This can be seen in Fig 2.1, where requesting the discriminative image regions for the class 'Dog' returns a large amount of activity around the dog's face, whereas the 'Seashore' activations cover the water surface.

(a) Original image (b) Ball (c) Dog (d) Seashore

Figure 2.1: Discriminative regions per class, images from [8]

[1] https://github.com/metalbubble/CAM

CAMs are generated by projecting the class activation scores of a convolutional neural network (CNN) back onto the last convolutional layer. Because convolutional layers have two spatial dimensions that correspond to the spatial dimensions of the image being classified, the activations of the final layer can be scaled up to fit on top of the original image (as in Fig 2.1). One precondition that a CNN classifier has to satisfy to produce CAMs is the use of a General Average Pooling (GAP) layer as a second-last layer and a softmax layer as a last layer. GAP layers retain spatial information and relative importance of features, whereas fully connected layers, which are employed by many popular CNN architectures, fail to do so. This requirement limits the number of classifiers at our disposal. As will be justified in the Methods chapter, the Resnet18[9] classifier is used in this paper.

To gain an understanding of how CAMs are created, the underlying calculations specified in [7] are described here. These calculations allow for the inference from class activation scores to CAMs. In other words, the process of taking the chance of an image being of class x and returning the discriminative image regions for class x.

- Let k be a feature unit in the last convolutional layer, i.e., a coloured layer in Fig 2.2.
- Let $f_k(x, y)$ be the activation of unit k at the spatial location (x, y).
- The result of performing global average pooling for a unit k is denoted by F^k. This is represented by a coloured dot in Fig 2.2.
- $F^k = \sum_{x,y} f_k(x, y)$
- The input of the final softmax layer is a combination of the globally pooled averages. The weight of each average is given by w_k in Fig 2.2. These weights signify the importance of a feature unit k for a class. Specifying the relevant class for each weight in the notation gives us w_k^c.
- Finally, when ignoring the softmax function, a class score is given by the weighted sum of all globally pooled averages: $S_c = \sum_k w_k^c F_k$.

Given these statements we can rewrite our class score as a summation of activations at spatial locations.

$$S_c = \sum_k w_k^c F_k = \sum_{x,y} \sum_k w_k^c f_k(x, y)$$

Separating the feature units k from this equation gives us.

$$S_c = \sum_{x,y} M_c(x, y) \qquad \text{where} \qquad M_c(x, y) = \sum_k w_k^c f_k(x, y)$$

Thus, the final class score can be derived from scores M_c at specific spatial locations. As these locations on the final convolutional layer can be sampled up to the size of the original image, we can build a visual map of regions that contribute significantly to class activation scores.

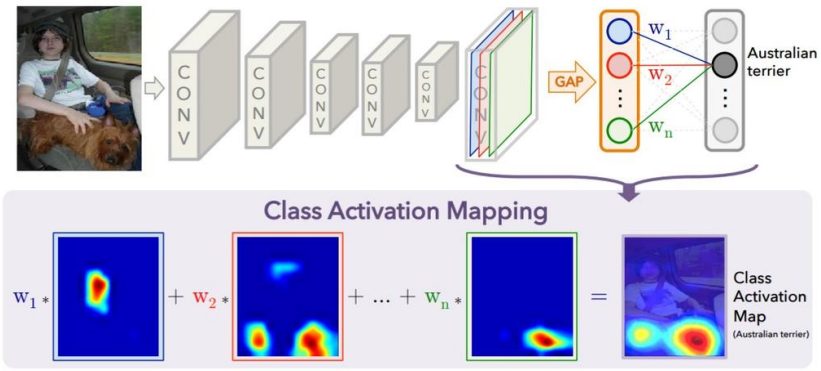
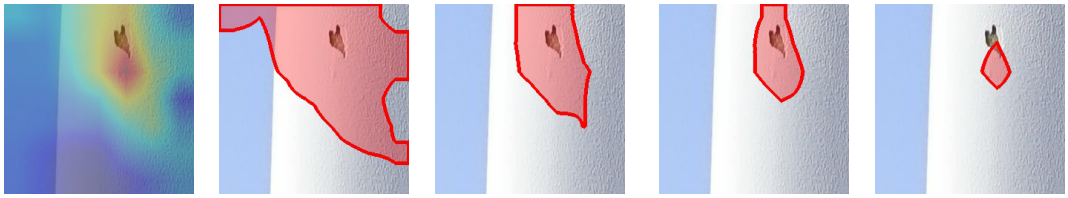


Figure 2.2: Generating a CAM from GAP activations, image from [7]

2.2 Mask R-CNN

Given that the Class Activation Mappings will allow for the generation of pixel-level training data, the next step in assembling a segmentation pipeline is choosing a strongly supervised segmentation algorithm. Such an algorithm could train on pixel-level data and alleviate several issues that plague CAMs. Because of the static sizes and 1:1 aspect ratio of convolutional layers, images with exceptionally large or small aspect ratios will yield distorted CAMs. Furthermore, determining which pixels of a CAM are to be included in a polygon or pixel-level annotation requires a threshold limit that has to be determined manually (Fig 2.3). Lastly, the center of high activations in CAMs are often poorly localised, as can be seen in Fig 2.3e. This deficiency arises due to the distortion of spatial dimensions throughout many layers of convolutions, meaning that a neuron in the final convolutional layer does not directly correspond with a pixel location in the original image, but rather to a local region. This effect is amplified when using many deep convolutional layers.



(a) CAM (b) 30% (c) 50% (d) 65% (e) 80%

Figure 2.3: CAM activity% threshold values

A strongly supervised algorithm can eliminate the need for any hard coded parameters for thresholding. A network such as Mask R-CNN determines the boundaries of objects based on features of the underlying pixels. This naturally also avoids problems with poorly localised segmentation, as the boundaries depend directly on image features instead of approximate regions of CAM activity. Images with extreme aspect ratios also pose no problem for Mask R-CNN as images do not have to be stretched to a 1:1 aspect ratio. Instead images are processed at the level of regions of interest (RoI) which can take on arbitrary aspect ratios.

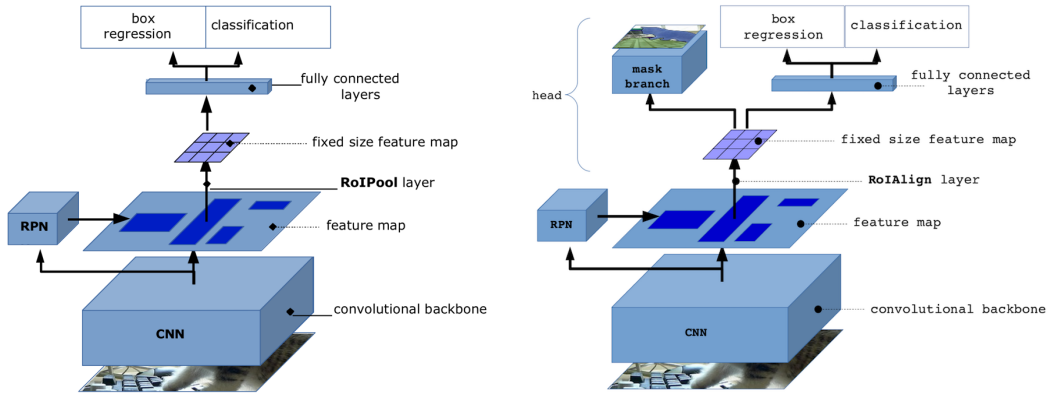
While choosing a strongly supervised algorithm, there was a preference for systems that were readily available, well documented and exhibited state of the art performance on large-scale data sets. Several networks were considered; among the most promising were FCIS, Masknet, Mask R-CNN, Polygon-RNN++ and PANet, which is essentially an improved Mask R-CNN. Table 2.1 specifies some of the relevant properties for each model.

	Code available[2]	Well documented	Published	mAP @ 0.5 COCO
Polygon-RNN++ [10]	no	yes	2018	45.5
FCIS [11]	yes	yes	2016	51.4
Masknet [12]	no	yes	2017	55.1
Mask R-CNN [4]	yes	yes	2017	58.1
PANet [13]	no	yes	2018	63.1

Table 2.1: Strongly supervised segmentation networks

[2]This applies to both the code for makings inferences as the code for training a network. Some networks such as Polygon-RNN++ have publicly available inference code, yet no means are provided to train the network on other classes.

After careful consideration Mask R-CNN was chosen, with its high performance and availability[3] being the decisive factors. Mask R-CNN is a network developed by Ross Girschink and his teams at Microsoft and Facebook AI Research. The network stems from a lineage of object detection networks which started with R-CNN [14], followed by the more precise and efficient Fast R-CNN [15] and Faster R-CNN [16] networks. These three networks are object detection networks, meaning that they were designed to detect an object in an image and return a bounding box that envelops the object. Mask R-CNN is the most recent addition to this family of networks. It improves upon Faster R-CNN by adding a mask prediction branch, as seen in Fig 2.4b. Although describing the precise function of each network component is not reasonable for the purpose of this work, it is important to note what functionality is added by this mask prediction branch. The branch allows for the network to be trained on pixel-level labels and for bounding box predictions to be refined to pixel segments. These capabilities make Mask R-CNN a strongly supervised instance segmentation algorithm.



(a) Faster RCNN (b) Mask R-CNN

Figure 2.4: Evolution of RCNN network family, images from [4]

2.3 Combining weakly and strongly supervised segmentation

The purpose of this work is not to demonstrate the performance of readily available techniques on a specific data set; rather, the goal was to show that two fundamentally different segmentation techniques could be combined to achieve 'the best of both worlds'. That is, the low-cost training data of weakly supervised methods with the robustness and learned features of strongly supervised methods. Rather than representing an academic curiosity, this approach was born out of necessity when Birds.ai desired the segmentation performance of strongly supervised methods whilst not wanting to use expensive pixel-level training data. Whether combining these two methods would yield decent results was uncertain. To the best of our knowledge there have been no published works on the subject yet.[4]

Linking the two methods will most likely cause poor performance if Mask R-CNN is not capable of learning a general model from the noisy CAM data; in other words, if the undamaged regions included in CAM polygons (Fig 2.3) inhibit Mask R-CNN from learning features required for segmenting damage. A poorly trained Mask R-CNN model would segment undamaged surfaces that share visual cues with undamaged surfaces in the training data. Thankfully, there are a few reasons why this may not be the case.

[3] https://github.com/matterport/Mask_RCNN
3

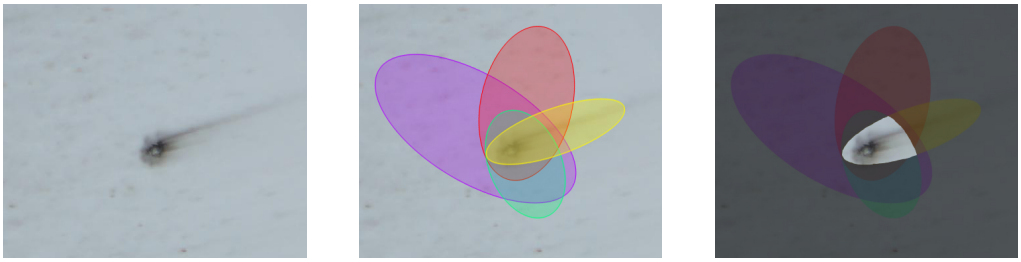
[4] The first 5 result pages for queries on Google scholar contained no relevant literature. The query strings consisted of combinations of the words: "weak(ly), strong(ly), (semi)supervised, segmentation, combine, combination".

Firstly, damaged surfaces are more distinctive than undamaged surfaces. That is to say, if there are enough images in the data set, features that we associate with damage will be significantly better predictors of damage than features we associate with undamaged surfaces. Dirt, smudges and dark colours predict the location of damage well, as CAM damage segments always contain damage with these features. Clean and evenly coloured surfaces might be present in the noisy data, but they predict many regions that are not damaged and are not in CAM damage segments. Therefore features from undamaged surfaces are inferior.

Secondly, damaged surfaces are seen consistently in the noisy CAM data, whereas undamaged surfaces are not. A visualisation of this is given in Fig 2.5, where 5 different CAMs all overlap on a damaged surface. Although this visualisation does not apply to this thesis, as the classifier used will always generate the same CAM for an image, it is a way of visualising how damaged surfaces with useful features will be consistently included, while undamaged surfaces with unhelpful features will appear sporadically.

Thirdly, given that a model does not overfit the data, only the truly generalisable features for segmenting damage will be retained. The truly generalisable features for the noisy CAM polygons are the damaged surfaces they contain. Mask R-CNN utilises weight-decay as a regularisation method to reduce overfitting.

Although these three point do not guarantee that Mask R-CNN will learn useful features, they reveal that combining the two methods is at the very least worthy of an attempt. As will be shown in the Results chapter, Mask R-CNN is in fact capable of learning useful features from the noisy CAM data.



(a) Original image (b) 5 different CAMs (c) Area most often seen

Figure 2.5: Consistency in noisy data

CHAPTER 3

Methods

3.1 Data

Birds.ai provided 4.094 high resolution images of wind turbines. These pictures originated from the inspection of 10 separate turbines. From these images, 250 images with image-level damage annotations were obtained and 280 of undamaged surfaces (Fig 3.1). 185 of these annotations were manually acquired by extracting local image regions around damaged areas. 75 images were found by using Birds.ai's API, which allowed for extracting images regions that were known to contain damage. 227 of the 250 images contained a single damaged surface, 23 contained two or more. To be able to test the performance of the pipeline, 50 damaged and 50 undamaged images were reserved for the test set. Each damaged image in the test set had a ground truth (gt) associated with it. The remaining images in the training set were augmented by mirroring them horizontally, vertically and diagonally. This made for a total of 800 damaged and 920 undamaged images in the training set.

Figure 3.1: Damaged and undamaged images from the training set

Birds.ai uses several labels to classify different types of damage; these include lightning strikes, oil spills and dirt. To maximise the amount of data available for this project, every damaged surface that had a clearly visible boundary and was restricted to a local image region was included. This means that faint damages that occupied large surfaces with no clear boundaries were excluded. An example of this is the washed off dirt in the second undamaged image in Fig 3.1.

3.2 Equipment

For the Restnet18 classifier a Pytorch[1]implementation from the standard Torchvision[2] package is used. For the Class Activation Mappings the implementation developed by the authors of [7] is used. Their code was published on Github[3] . This implementation is written in Python[4] and communicates directly with classifiers written in Pytorch. For Mask-RCNN an open source implementation from Github[5] is used. This implementation is written in Keras[6] and TensorFlow[7]. Training Mask R-CNN and Resnet took place on an Ubuntu 16.04 desktop using an Nvidia GeForce GTX 1060 6GB graphics card. To create ground truth segments for damaged images in the test set, Adobe Photoshop CS6[8] was used. Photoshop was also used for editing and post-processing most of the images that appear in this report. Python3.6 was used for all non-GPU batch processes, such as augmenting the training data by mirroring.

3.3 Pipeline

The segmentation pipeline is shown in Fig 3.2, where each blue arrow represents the flow of data into a processes. The pipeline begins with a classifier that is trained to distinguish damaged from undamaged images. A Resnet18 network was chosen as it is the network with the least convolutional layers available in the Torchvision library that uses a global average pooling layer. As noted in [7] and paragraph 2.2, the less convolutional layers a network has, the weaker the spatial distortion is in the CAMs. A base learning rate of 0.001 was used, with a exponential decay of 0.1 every 90 epochs. Each epoch contained 400 training steps, for which a batch size of 10 was used. The network was trained for 300 epochs.

The trained classifier then generates 800 CAMs of the damaged images in the training set. It may seem as though this is a bad practice; the same data that is used for training the classifier is now also used for generating the CAMs. Whether this is harmful depends on what the pipeline is used for. If it is a purely practical undertaking and the goal is to attain the highest possible segmentation performance with a limited amount of data, this will be the best course of action. If the goal is to investigate the robustness and reusability of a trained classifier, it is bad practice. This thesis studies the former case.

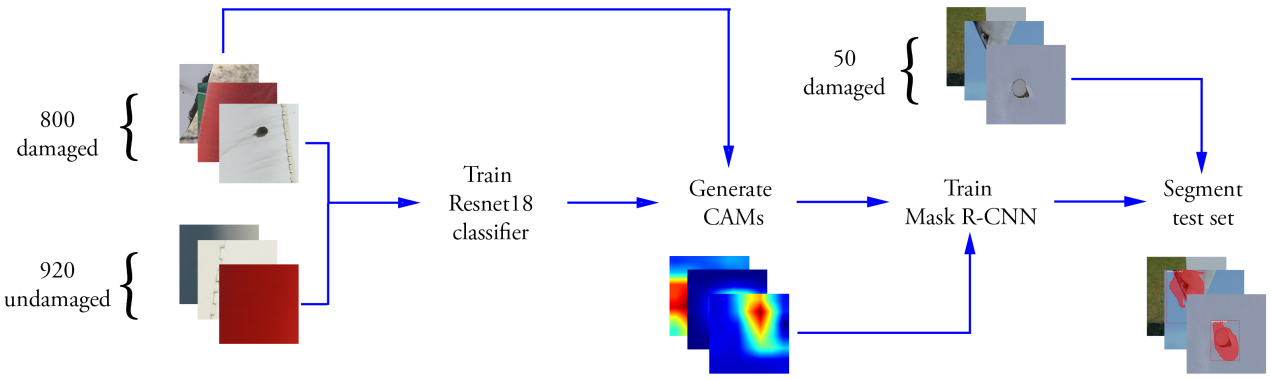


Figure 3.2: Sequence of actions to perform segmentation

A threshold of 65% is applied to the 800 CAMs to obtain pixel-level annotation for training

[1]https://pytorch.org/
[2]https://github.com/pytorch/vision
[3]https://github.com/metalbubble/CAM
[4]https://www.python.org/
[5]https://github.com/matterport/Mask_RCNN
[6]https://keras.io/
[7]https://www.tensorflow.org/
[8]https://www.adobe.com/products/photoshop.html

Mask R-CNN (2.3d). This threshold was chosen manually after trial and error optimisation of the tradeoff between including damage and including undamaged surfaces.

Mask R-CNN is then trained on the pixel-level annotations and predictions are generated for the 50 damaged images in the test set. Training entails 100 epochs, each of which contains 1000 training steps. The regions of interest per image is limited to 256 and a batch size of 1 is used. Input image resolution are restricted to 1024 and a weight decay of 0.0001 is utilised.

CHAPTER 4

Results

The performance of the final segmentation network depends on that of all preceding pipeline components. Because of this there will be a quantitative analysis of all components. A qualitative analysis of 5 randomly selected images in the test set can be found in Fig 4.4.

4.1 Classifier

To produce high quality CAMs, a classifier must learn features that are discriminative for a class. Classifier performance demonstrates to what extent the model can discriminate classes. Therefore, if the test set has an adequate size, a good performance indicates that discriminative features have been learned. A popular format used to describe the performance of a classifier is a confusion matrix (Table 4.1).

		Classified as	
		Broken	Unbroken
Actual label	Broken	48	2
	Unbroken	3	47

Figure 4.1: Confusion matrix

From a confusion matrix certain performance metrics can be derived, such as accuracy and F_1 score.

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} = \frac{95}{100} = 0.95$$

$$F_1 = \frac{2TP}{2TP+FP+FN} = \frac{96}{101} = 0.95$$

These high outcomes are not surprising given the simplicity of the classification task. A data set with a total of 1720 images is used for the binary classification of well defined classes with a sufficiently complex CNN that is trained to convergence.

4.2 Class Activation Mappings and Mask R-CNN

Measuring the extent to which the CAMs and Mask R-CNN segments match ground truths requires a metric known as intersection over union (IoU). IoU is a convenient metric for this task as it penalises both segments that include only small regions of the ground truth, as well

as segments that cover vast regions in which only a small subsample covers the ground truth. A visual description of IoU is given in Fig 4.2 and Fig 4.3.

(a) Gold standard (b) Prediction (c) Intersection (d) Union

Figure 4.2: IoU visualisation

All segments that do not intersect a ground truth have an IoU of 0. Using this assumption when calculating the average IoU for both methods results in the values displayed in Table 4.1

	Segment intersects gt	Segment covers gt	Average IoU
Mask R-CNN	0.96	0.92	0.12
CAMs	0.88	0.82	0.06

Table 4.1

Both methods have an average IoU that is far lower than that of state of the art strongly supervised segmentation methods, as can be inferred from the final column of Table 2.1, where more than half of the segments have an IoU equal to or greater than 0.5. Despite this, it is essential to acknowledge that no pixel-level data was used for training. Weakly supervised methods and strongly supervised improvements thereof should not solely be compared to pixel-level competitors but also be considered methods unto themselves with their own set of applications. Also, the high intersection and coverage percentage indicate that the methods are effectively damage detectors. Very few of the predicted segments completely miss damaged surfaces. Because of this property a user can select an image region with a bounding box and have the damaged surface segmented by the Mask R-CNN network with a certainty of 96%, albeit a non-optimal segment.

Mask R-CNN doubled the average IoU of CAMs, showing that the combination of strongly and weakly supervised methods can lead to an increase of performance, a feat that to the best of our knowledge has never been demonstrated prior to this thesis. Ideally Mask-RCNN would only learn features that belong to damaged surfaces. Although this potentially could have caused the doubling of the average IoU relative to CAMs, Fig 4.4 shows that there are still large regions of undamaged surfaces included in the Mask R-CNN segments.

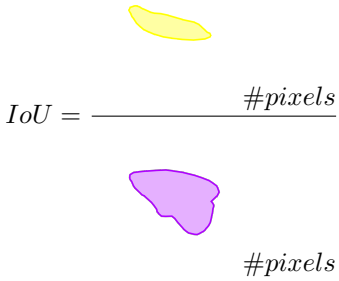


Figure 4.3: IoU calculation with shapes from Fig 4.2

Mean average precision (mAP) is another widely adopted metric for measuring the performance of segmentation algorithms. It relies on the IoU metric and can be calculated as follows.

$$\text{mAP @ } x \text{ IoU} = \frac{1}{classes} \sum_{c \in classes} \frac{\#TP(c)}{\#TP(c)+\#FP(c)} \text{ @ } x \text{ IoU}$$

Here the x IoU determines how strict one is when considering how tightly a segment has to fit the ground truth to count as a true positive. Because there is only a single class being segmented in this project, the accuracy is equal to the mean accuracy. The formula can therefore be simplified to AP.

$$\text{AP @ } x \text{ IoU} = \frac{\#TP}{\#TP+\#FP} \text{ @ } x \text{ IoU}$$

The AP of Mask R-CNN and CAMs can be found in Table 4.2

	AP @ 0.05	AP @ 0.10	AP @ 0.20	AP @ 0.30
Mask R-CNN	0.82	0.62	0.40	0.20
CAMs	0.76	0.40	0.20	0.10

Table 4.2

As is the case with the average IoU, the performance is less than that of strongly supervised methods trained op pixel-level data. Noticeably, 20% of all Mask R-CNN segments have an IoU equal to or greater than 0.30. An IoU of 0.30 is widely considered to be an acceptable amount of overlap, with the authors of [14] using an IoU of 0.3 as a threshold for considering a sample positive.

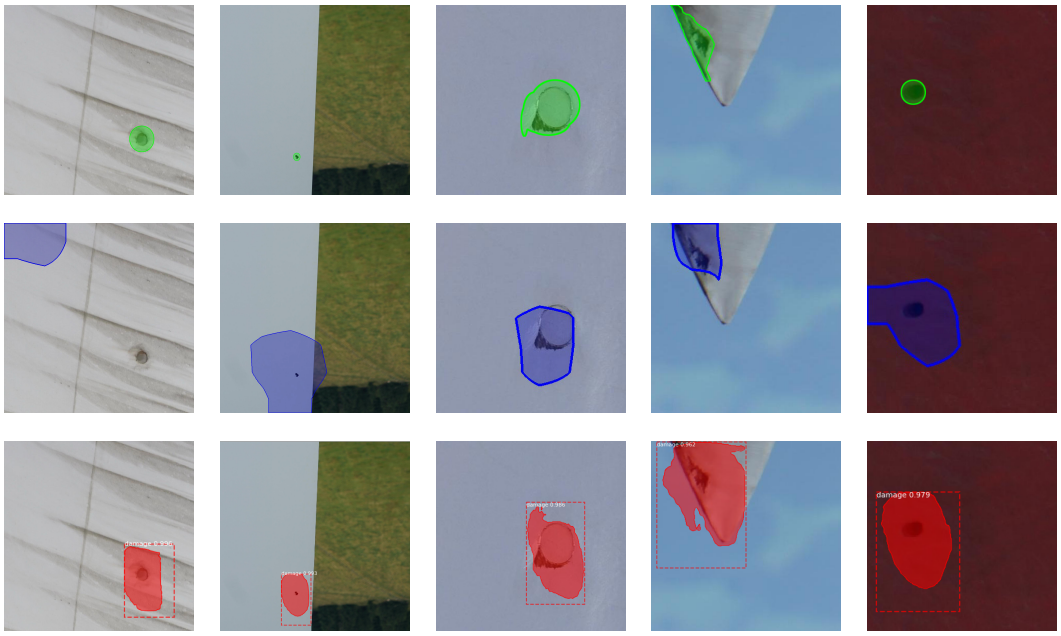


Figure 4.4: Top row: Ground truths. Middle row: CAMs. Bottom row: Mask R-CNN segments.

CHAPTER 5

Discussion

One of the main theoretical motivators for appending a Mask R-CNN network to the pipeline was to cover less undamaged surfaces than CAMs. Despite a better average IoU score, there are still large undamaged surfaces covered by Mask R-CNN segments as can be seen in Fig 4.4. This beckons further experimentation with stricter regularisation, such as stronger weight decay than the standard 0.0001 or less training epochs for Mask R-CNN. Regularisation would place constraints on the features the network could retain, causing weaker features of undamaged surfaces to be discarded.

The threshold for CAMs were chosen manually after trial and error optimisation of the tradeoff between including damage and including undamaged surfaces. Although this has produced acceptable pixel-level data, an automatic optimisation based on IoU would most probably increase the quality of CAMs segments. Other weakly supervised methods with a more sophisticated design than CAMs should also be considered in future research, such as the one described in the recently published [6].

During this thesis there was a limited supply of data for testing the potential of combining weakly and strongly supervised methods. While a training set of 200 damaged images seems acceptable, the types of damage does vary, from punctures to oil spills and peeling paint. Experiments on large-scale data sets would give a better indication of the improvements one could expect from appending a strongly supervised method to the pipeline.

Finally, the pipeline used in this thesis is not a unified component with a single training and inference procedure, but rather a collection of separate components, which renders the pipeline difficult to use. Future research could investigate the possibility of passing the output of weakly supervised methods as a feature map or layer in a CNN.

CHAPTER 6

Conclusions

In this thesis a novel combination of weakly and strongly supervised segmentation methods was proposed for segmenting damage on wind turbines. A strongly supervised Mask R-CNN model was trained on pixel-level data generated by weakly supervised CAMs. Segments created by the Mask R-CNN network were significantly better than those produced by CAMs, achieving an average precision of 20% with an IoU threshold of 0.30, effectively doubling the performance of the CAMs. Both methods covered the majority of the ground truth segments in the test set. Because of this the final model enables users to create polygons that cover wind turbine damage with a certainty of 92% with only two mouse clicks. More research will be needed to determine the full potential of this approach, especially experimentation on large-scale data sets. We hope that this approach will serve as a motivator for future research in combining weakly and strongly supervised methods.

Bibliography

[1] Sang Uk Lee, Seok Yoon Chung, and Rae Hong Park. "A comparative performance study of several global thresholding techniques for segmentation". In: *Computer Vision, Graphics, and Image Processing* 52.2 (1990), pp. 171–190. URL: https://www.sciencedirect.com/science/article/pii/0734189X9090053X?via%3Dihub.

[2] Arthur Robert Weeks and G Eric Hague. "Color segmentation in the HSI color space using the K-means algorithm". In: *Nonlinear Image Processing VIII*. Vol. 3026. International Society for Optics and Photonics. 1997, pp. 143–155. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/3026/0000/Color-segmentation-in-the-HSI-color-space-using-the-K/10.1117/12.271117.short?SSO=1.

[3] Liu Jianzhuang, Li Wenqing, and Tian Yupeng. "Automatic thresholding of gray-level pictures using two-dimension Otsu method". In: *Circuits and Systems, 1991. Conference Proceedings, China., 1991 International Conference on*. IEEE. 1991, pp. 325–327. URL: https://ieeexplore.ieee.org/abstract/document/184351/.

[4] Kaiming He et al. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

[5] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *CoRR* abs/1511.00561 (2015). arXiv: 1511.00561. URL: http://arxiv.org/abs/1511.00561.

[6] Yanzhao Zhou et al. "Weakly Supervised Instance Segmentation using Class Peak Response". In: *arXiv preprint arXiv:1804.00880* (2018). URL: http://openaccess.thecvf.com/content_cvpr_2018/CameraReady/0974.pdf.

[7] Bolei Zhou et al. "Learning deep features for discriminative localization". In: *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE. 2016, pp. 2921–2929. URL: http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf.

[8] Albert Jimenez, Jose M Alvarez, and Xavier Giro-i Nieto. "Class-weighted convolutional features for visual instance search". In: *arXiv preprint arXiv:1707.02581* (2017). URL: https://arxiv.org/pdf/1707.02581.pdf.

[9] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[10] David Acuna et al. "Efficient Interactive Annotation of Segmentation Datasets With Polygon-RNN++". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 859–868. URL: http://openaccess.thecvf.com/content_cvpr_2018/papers/Acuna_Efficient_Interactive_Annotation_CVPR_2018_paper.pdf.

[11] Yi Li et al. "Fully convolutional instance-aware semantic segmentation". In: *arXiv preprint arXiv:1611.07709* (2016). URL: http://openaccess.thecvf.com/content_cvpr_2017/papers/Li_Fully_Convolutional_Instance-Aware_CVPR_2017_paper.pdf.

[12] JULIANO PINTO. "Masknet: An Instance Segmentation Algorithm". In: (2017). URL: http://publications.lib.chalmers.se/records/fulltext/250417/250417.pdf.

[13] Shu Liu et al. “Path aggregation network for instance segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8759–8768. URL: http://openaccess.thecvf.com/content_cvpr_2018/CameraReady/2784.pdf.

[14] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.pdf?spm=5176.100239.blogcont55892.8.pm8zm1&file=Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.pdf.

[15] Ross Girshick. “Fast r-cnn”. In: *arXiv preprint arXiv:1504.08083* (2015). URL: http://openaccess.thecvf.com/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf.

[16] Ross Girshick. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.